VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
No department name is given

Absolvování individuální odborné praxe

Individual Professional Practice in the Company

2016 Michal Kopečný

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Michal Kopečný**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Absolvování individuální odborné praxe
Individual Professional Practice in the Company

Jazyk vypracování: čeština

Zásady pro vypracování:

1. Student vykoná individuální praxi ve firmě: EParts Group s.r.o.
2. Struktura závěrečné zprávy:
   a) Popis odborného zaměření firmy, u které student vykonal odbornou praxi a popis pracovního zařazení studenta.
   b) Seznam úkolů zadaných studentovi v průběhu odborné praxe s vyjádřením jejich časové náročnosti.
   c) Zvolený postup řešení zadaných úkolů.
   d) Teoretické a praktické znalosti a dovednosti získané v průběhu studia uplatněné studentem v průběhu odborné praxe.
   e) Znalosti či dovednosti scházející studentovi v průběhu odborné praxe.
   f) Dosažené výsledky v průběhu odborné praxe a její celkové zhodnocení.

Seznam doporučené odborné literatury:

Podle pokynů konzultanta, který vede odbornou praxi studenta.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **doc. Ing. Zdeněk Sawa, Ph.D.**

Konzultant bakalářské práce: Mgr. Jiří Glinz

Datum zadání: 01.09.2015

Datum odevzdání: 29.04.2016

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 29. dubna 2016 ..... Kopčný .....

Souhlasím se zveřejněním této bakalářské práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v bakalářských programech VŠB-TU Ostrava.

V Ostravě 29. dubna 2016

Abstrakt

Tato bakalářská prace byla vypracována formou inviduální odborné praxe ve firmě Eparts Group s.r.o. V této práci je shrnut popis firmy včetně pracovního postupu, mých úkolů a postupů použité při jejich řešení. Na závěr provádím zhodnocení nabytých zkušeností a dovedností a míry splnění zadaných úkolů. Mým dlouhodobým zadáním byl vývoj skladového hospodářství a automatizace nasazení E-shopu. Krátkodobé úkoly se týkaly oprav chyb nebo rozšiřování funkcionalit E-shopu.

Klíčová slova: Eparts Group s.r.o., E-shop, skladové hospodářství, PHP, Node.js, AngularJS, Javascript, TypeScript, Nette

Abstract

This Barchelor thesis was made as invidual professional practice in Eparts Group s.r.o. In this thesis, I summarized description of company including their workflow, my assigned tasks and their solutions. I conclude the thesis by evaluating my gained experience and skills and the rate of completion of asigned tasks. My long assigments were development of warehouse management system and automatizating deployment of E-shop. My short term goals were fixing bugs and expanding functions of E-shop.

Key Words: Eparts Group s.r.o., E-shop, warehouse management, PHP, Node.js, AngularJS, Javascript, TypeScript, Nette

Obsah

Seznam použitých zkratek a symbolů

ACL	–	Access List
AJAX	–	Asynchronous JavaScript And Xml
API	–	Application Programming Interface
DB	–	Database
DI	–	Dependency Injection
DOM	–	Document Object Model
E2E	–	End To End
HTML	–	HyperText Markup Language
HTTP	–	HyperText Transfer Protocol
JS	–	Javascript
LAMP	–	Linux Apache MySQL PHP
OOP	–	Object Oriented Programming
PHP	–	PHP: Hypertext Preprocessor
REST	–	Representational State Transfer
UI	–	User Interface
XHR	–	XML Http Request

Seznam obrázků

1 Úvod

Tématem této práce je vykonání odborné praxe ve firmě Eparts Group s.r.o. V úvodu práce uvedu stručnou historii firmy, popíšu pracovní postup vývoje, který firma používá a stručně uvedu technologie používané firmou. Zadané úkoly, které mi byly zádany, se týkali vývoje a udržby dvou řešení.

První skupina úkolů se týkala E-shopu a byly zaměřeny na údržbu a rozšiřování funkčnosti aplikace ale součástí byl i úkol s nízkou prioritou a to návrh a případný vývoj systému pro automatizovanou instalaci nebo aktualizaci pomocí ovládacího panelu a GIT repozitáře.

Dalším úkolem byl návrh architektury a struktury aplikace pro skladové hospodářství. Pro každé řešení jsou vybrány významné úkoly, u kterých je stručný popis problému a jeho řešení.

V závěru provedu zhodnocení stupně splnění jednotlivých úkolů, nastíním předpokládaný průběh dalšího vývoje a navrhnu chybějící fukcionalitu nebo část pracovního postupu, která by měla být v budoucnu zařazena do seznamu úkolů. Dále provedu zhodnocení získaných dovedností a zkušeností a uvedu znalosti a dovednosti, které jsem si musel doplňovat v průběhu praxe.

2 Firma

2.1 Historie firmy

Společnost EParts Group s. r. o. byla založena v roce 2011 za účelem maloobchodního prodeje autodílů a autodoplňků prostřednictvím internetového obchodu.

Po vyzkoušení několika hotových systémů se EParts Group s. r. o. v roce 2012 rozhodla přejít na vývoj vlastního řešení na bázi katalogu náhradních dílů TecDoc a zároveň změnit své zaměření z prodeje autodílů a autodoplňků na vývoj E-shopového řešení, jeho pronájem a poskytování služeb pro prodejce autodílů.

2.2 Vývoj

Vývojový tým společnosti je tvořen třemi lidmi a vývoj probíhá v několika fázích.

1. **Plánovací fáze** v této fázi probíhá diskuze a analýze požadavku, jeho členění na úkoly a přiřazování úkolů jednotlivým členům.
2. **Implementační fáze** v této fázi probíhá implementace a testování funkčnosti.
3. **fáze křížového začlenění** v této fázi člen týmu, který se nepodílel na daném úkolu prochází vytvořený kód, provádí jeho kontrolu a posuzuje, zda je kód dostatečně srozumitelný.
4. **fáze nasazení** Tato fáze spočívá ve vydání nové verze a její nasazení na produkční prostředí.

2.3 Zařazení

Firma mě zařadila do vývojového týmu, kde moje pracovní náplň spočívala ve vývoji a opravování chyb v E-shopovém řešení a započetí implementace řešení skladového hospodářství.

3 Používané technologie

Jako databáze je firmou používána relační databáze MySQL komunitní edice. MySQL je rozšířená open source relační databáze, jejihž největší výhodou je její rozšířenost a možnost provozu na vlastním serveru.

3.1 E-shop

E-shop řešení je postavené na LAMP stacku za použití PHP frameworku Nette a databázové abstrakční vrstvy Dibi. Pro klientskou část je použita knihovna JQuery a framework Bootstrap.

PHP je populární všeobecný skriptovací jazyk, který je zaměřený zejména pro vývoj webu.[3]

Nette je open source PHP framework s plným využitím objektů (OOP). Jeho autorem je David Grudl. Ačkoliv vznikl už v roce 2004, teprve v roce 2009 byl uvolněn jako open source a zpřístupněn veřejnosti. Jeho licence, která vychází z BSD, patří k těm nejvolnějším.[4]

Dibi je databázová abstraktní vrstva vytvořená Davidem Grudlem, jejihž cílem je zjednodušit zápis SQL příkazů, eliminace SQL chyb pomocí zjednodušení zápisu a přenositelnost mezi databázovými systémy.[5]

JQuery je rychlá, malá a funkcemi nabitá JavaScriptová knihovna, která usndaňuje procházení a manipulaci HTML dokumentu, zpracování událostí, animace a AJAX požadavků a zároveň sjednocuje funkcionalitu napříč prohlížeči.[6]

Bootstrap je elegantní, intuitivní a výkonné front-end framework pro rychlejší a snadnější vývoj webových aplikací. Jeho autory jsou Mark Otto a Jacob Thornton, a udržován hlavním týmem s masivní podporou a zapojením komunity.[7]

3.2 Skladové hospodářství

3.2.1 Server

Serverová část je postavena na principu REST služby za použití Node.js a frameworku Restify. Jako uložiště bylo zvoleno MySQL, které již firma používála a má vybudovanou architekturu.

Node.js je asynchronní, na událostech založené běhové prostředí využívající V8 JavaScript engine. Node.js je velmi vhodný na aplikace, které provádějí spoustu operací, které způsobují čekání na výsledek (XHR požadavek, čtení souboru, síťová kominikace), protože během čekání na výsledek operace může Node.js provádět další operace dokud nedorazí událost, která oznamuje hlavnímu vláknu, že výsledek operace je k dispozici.[9]

Restify je node.js modul vytvořený ke tvorbě korektních REST webových služeb. Restify je silně inspirovanáno modulem express ale není tak široce zaměřený jako express.[10]

3.2.2 Klient

Klientská část je postavena na frameworku angular.js a UI komponent angular Material.

Angular.js je JavaScriptový framwork, který využívá HTML ke strukturizaci a rozšiřuje jeho možnosti pomocí direktiv. Angular.js zastřešuje všechnu manipulaci s DOM, AJAX a duální binding (oboustrané propojení modelu a view). [8]

Angular Material Projekt angular material je referenční implementací Google's Material Design specifikace a obsahuje množinu znovupoužitelných, důkladně otestovaných a přístupných komponent uživatelského rozhraní založených na material designu. [11]

Typescript rozšiřuje JavaScript o typovost a kompilační fází. To umožňuje definovat rozhraní, provádět statickou kontrolu při kompilaci a usnadňuje refactoring kódu. [13]

4 Zadané úkoly

V následujících odstavcích popíšu vyznamné úkoly, na kterých jsem pracoval sám nebo kde množství práce, kterou jsem na úkolu odvedl, je dostatečně významné.

4.1 E-shop

První úkol se týká již existujícího řešení firmy, E-shopu. Nejvíce času, který byl určen, bylo využito k opravám chyb a vývoji nových funkcí.

4.1.1 Hledání a oprava chyby v importu dat

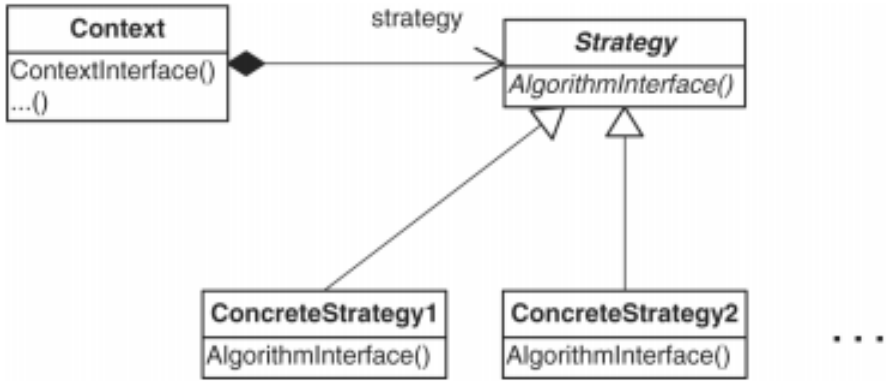
Vzhledem k důležitosti správnosti chodu importu dat se tohoto úkolu účastnil celý tým. Řešení jsme začali rychlým rozborem importovacího procesu a sestavováním seznamu míst, které by byly náchylné k způsobení pádu procesu. Po kontrole všech míst jsem chybu lokalizovali na poslední krok importu, který spočívá v kopírování dočasné tabulky do živé tabulky. Chybu způsoboval špatný formát katalogového čísla v importu, který způsobil selhání referenční integrity a ukončení procesu s chybou.

Pro vyhledání špatných záznamů jsem vytvořil databázovou proceduru, která za cenu výrazného snížení rychlosti pomocí kurzorů dokáže objevit všechna neplatná katalogová čísla.

Pro zabezpečení procesu před špatnými katalogovými čísly jsem implementovali opakovací frontu, která každou dávku, která neprojde, zařadí do opakování. Opakovací fronta je nakonec zpracována jednotlivě a tím oddělí neplatná čísla od čísel, která byla v dávce s neplatným číslem. Po provedení tohoto procesu zůstanou ve frontě neplatná čísla, která jsou vypsány ve stavové zprávě o importu. Toto umožňuje člověku, který spravuje import, provést nápravu a spustit import s opravenými daty.

4.1.2 Implementace SMS brány

V rámci tohoto úkolu jsem byl pověřen zjistit možnosti SMS bran a zvážit použití vlastního řešení, které nakonec nebylo implementováno z důvodu nutnosti mít pro každého klienta vlastní telefonní číslo. Použití SMS brány umožňuje nastavení identifikaci na název a proto byla SMS brána mnohem lepší výběr. Jedním z požadavků je modulárnost a pro splnění tohoto požadavku je nutné do DI kontejneru zaregistrovat všechny třídy, na které existuje závislost. Nesplnění závislosti zastaví běh aplikace a je navrácen HTTP kód 500 (Internal Server Error). Řešením tohoto problému bylo použití návrhového vzoru strategie, který spočíval v registraci závislosti na rozhraní a implementací komunikace s SMS bránou a vytvořením třídy, která imlementuje rozhraní ale neprovádí žádnou komunikaci. Jednotlivým instancím aplikace je následně v jejich konfiguraci vytvořena instance příslušné SMS brány a přidána do DI kontejneru. Použití tohoto vzoru zajišťuje, že DI je schopné splnit všechny své závislosti a zároveň umožnit rozdílnou konfiguraci jednotlivých instancí aplikace.



Obrázek 1: Návrhový vzor strategie [12]

4.1.3 Tvorba exportu pro služby Heureka.cz a Zboží.cz

Pro zlepšení nalezitelnosti E-shopu klientů zákazníky jsem byl pověřen vytvořením mechanismu pro vytváření exportu pro porovnávače. Vzhledem k množství zboží, které může být označeno pro export a omezení, vztahující se ke využití katalogu TECDOC a náročnosti procesu na paměť, bylo nutné generování provádět postupně. Z tohoto důvodu byl export rozdělen do 3 fází.

1. **fáze přírůstku do cache** Pro uložení zpracovaných produktů určených k exportu byl využit cache mechanismus Nette, kdy na začátku každého cyklu je vybráno N (N je číslo získano měřením nad daty určenými k testování) produktů, které nebyly dosud zpracovány a jsou cachovány. Po dokončení dojde ke spuštění 2. fáze.

2. **fáze tvorby XML souboru** K tvorbě XML souboru bylo možné použít XML knihovnu nebo soubor generovat ručně. Po vyzkoušení několika XML knihoven vyplynulo, že knihovny celý XML soubor udržují v pamětí a při větším objemu produktů by mohlo docházet k pádům procesu, způsobenými nedostatkem dostupné paměti. Vzhledem k této paměťové náročnosti a jednoduché struktury XML souboru bylo nakonec implementováno ruční generování souboru. Pro zvýšení spolehlivosti se vytváří nový soubor, který v případě úspěšného provedení operace nahradí původní. Pokud by došlo k selhání zůstane v platnosti původní soubor a správci je oznámena chyba.

3. **fáze odeslání XML souboru** Pro tuto fázi je v Nette vytvořena URL vedoucí na XML soubor, který si vyžádá Heureka.cz nebo Zboží.cz

4.1.4 Předělání výběru dopravy

Starý systém pro výběr doručení objednávky, který byl vytvořen jako výběr z roletky (select box), nebyl dostatečně flexibilní a rozšiřitelný. Při implementaci služby pro doručování do odběrných míst a připravy integrace s platebnímy branami jsme narazili na tato omezení. Jako řešení jsme rozhodli systém přepsat do modulární podoby, kde každá metoda platby je třída řešící své zpracování. Pro způsoby dopravy bylo zapotřebí vyřešit požadavek na vložení HTML a JS kódu. Toho bylo dosaženo využitím kategorií z původního systému a využití podmíněné kompilace šablonového systému Latte.

Další problém bylo dynamická aktivace prvků zavíslá na aktuálním výběru, indikace platností kombinací a zobrazení ceny při úplném výběru. Indikaci kombinací a výpočet ceny jsem dosáhl vytvořením komponenty pro přenos metadat a stavového automatu napsaného v JavaScriptu. Pro zobrazení prvků relevantních k aktuálnímu výběru jsem využil metadata a jednoduchou switch konstrukci s JQuery obsluhující zobrazování a skrývání prvků.

Po dokončení převodu na nový systém byl dalším úkolem provést rozšíření fukcionality. Klienti vznesli požadavek na nastavení cen v závislosti na hodnotě objednávky nebo případné úplné vynechání kombinace a zobrazování v závislosti na čase. Pro implementaci jsem využil funkci pro výpočet ceny, do které jsem přidal kontrolu validity a při nesplnění požadavků validity dojde k vyvolání vyjímky, která způsobí vyřazení ze seznamu dostupných kombinací.

4.1.5 Automatické nasazení

Návrh a implementace systému pro automatizované nasazení byl úkol s nízkou prioritou. Součástí úkolu bylo analyzovat současný způsob nasazení aplikace, který se provádí ručně v nepravidelných intervalech, navrhnout a v ideálním případě implementovat systém pro automatizovanou instalaci a aktualizaci. Bohužel tento úkol zůstal ve fázi návrhu.

Návrh, který jsme vypracovali, využivá systému štítků verzovacího systému GIT. Z hlediska implementace je třeba přidat vytvořit SSH klíč pro přístup ovládacího panelu pro E-shop a naprogramovat komunikaci ovládacího panelu s GITem. Po stažení projektu z gitu provede vytvoření složky s novou instalací a spustí instalační (nebo aktualizační) skript, který instalaci připraví pro chod. Následně provede zálohu databáze a aplikuje případné změny databázové struktury. Po dokončení těchto kroků je administrátor vyzván k potvrzení aktualizace. Po potvrzení této aktualizace dojde k přepsání aktivní složky na novou verzi.

4.1.6 Další vývoj

Z hlediska dalšího vývoje bude E-shop vyvíjen podle požadavků klientů, aplikace bude profilována a zrychlována a dojde k implementaci automatizovaného nasazení. Dále se po uvedení skladového hospodářství do provozu dojde k propojení těchto řešení.

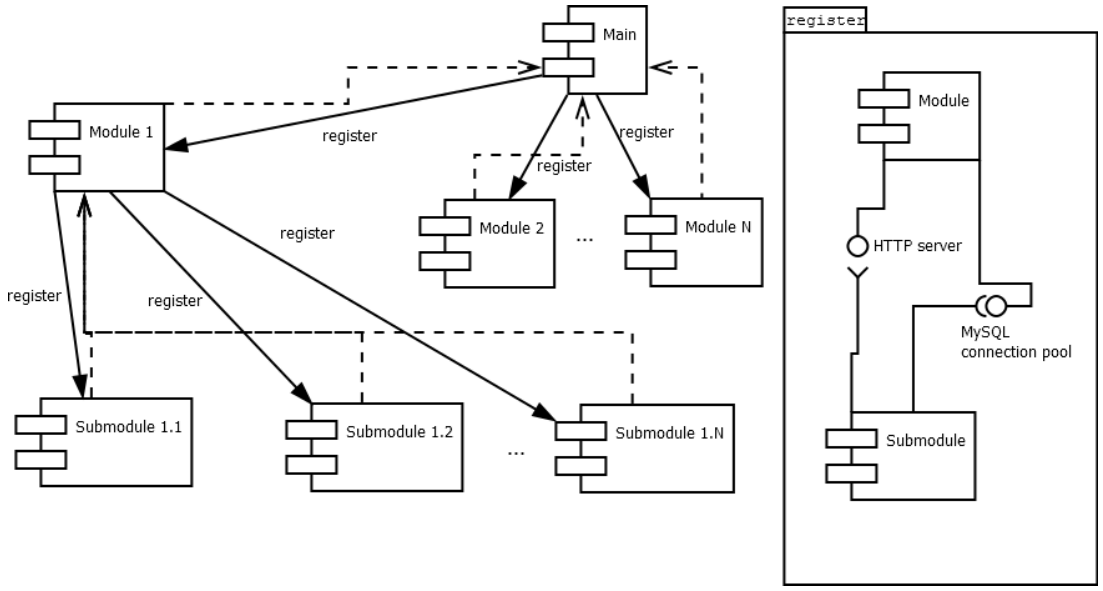
4.2 Skladové hospodářství

Druhým úkolem byl návrh a vývoj řešení skladového hospodářství. Součástí tohoto úkolu bylo rozhodnutí o použité architektuře, použitých jazyků a frameworků a následný vývoj tohoto řešení. Vzhledem k tomu, že skladové hospodářství bude používano v dlouhých časových intervalech a tím se zvyšují výhody, které by vyplynuly z použití REST serveru a aplikace běžící u klienta, bylo zvoleno použití Node.js poskytující REST službu na serverové straně a Angular.js frameworku na straně klienta. Jako databáze bylo zvoleno MySQL z důvodu již vybudované a používané architektury.

4.2.1 Server

4.2.1.1 Struktura aplikace Aplikace se skládá ze dvou hlavních částí. První částí je inicializační soubor. Jeho cíle jsou vytvoření worker procesů, vytvoření MySQL poolu a registrace modulů do REST serveru. Registrace modulu probíhá přidáním modulu a následným zavoláním jeho veřejné funkce register.

Druhou částí jsou moduly. Tyto moduly musí poskytovat funkci register s parametry pro předání HTTP serveru, použitého pro zaregistrování REST služeb a MySQL poolu pro komunikaci s databází. Struktura umožňuje i další členění modulu do podmodulů.



Obrázek 2: Modularita serveru skladového hospodářství

Každý modul je psán tak, aby neměl závislost na žádném dalším modulu a umožnil tak případné rozdělení aplikace a nasazení load balanceru.

4.2.1.2 REST API

Při tvorbě REST API bylo hlavně zohledněno umístění modulu ve stromě modulů, jednoduchost a expresivita cest. Expresivita cest byla zvýšena definováním této konvence využití HTTP sloves.

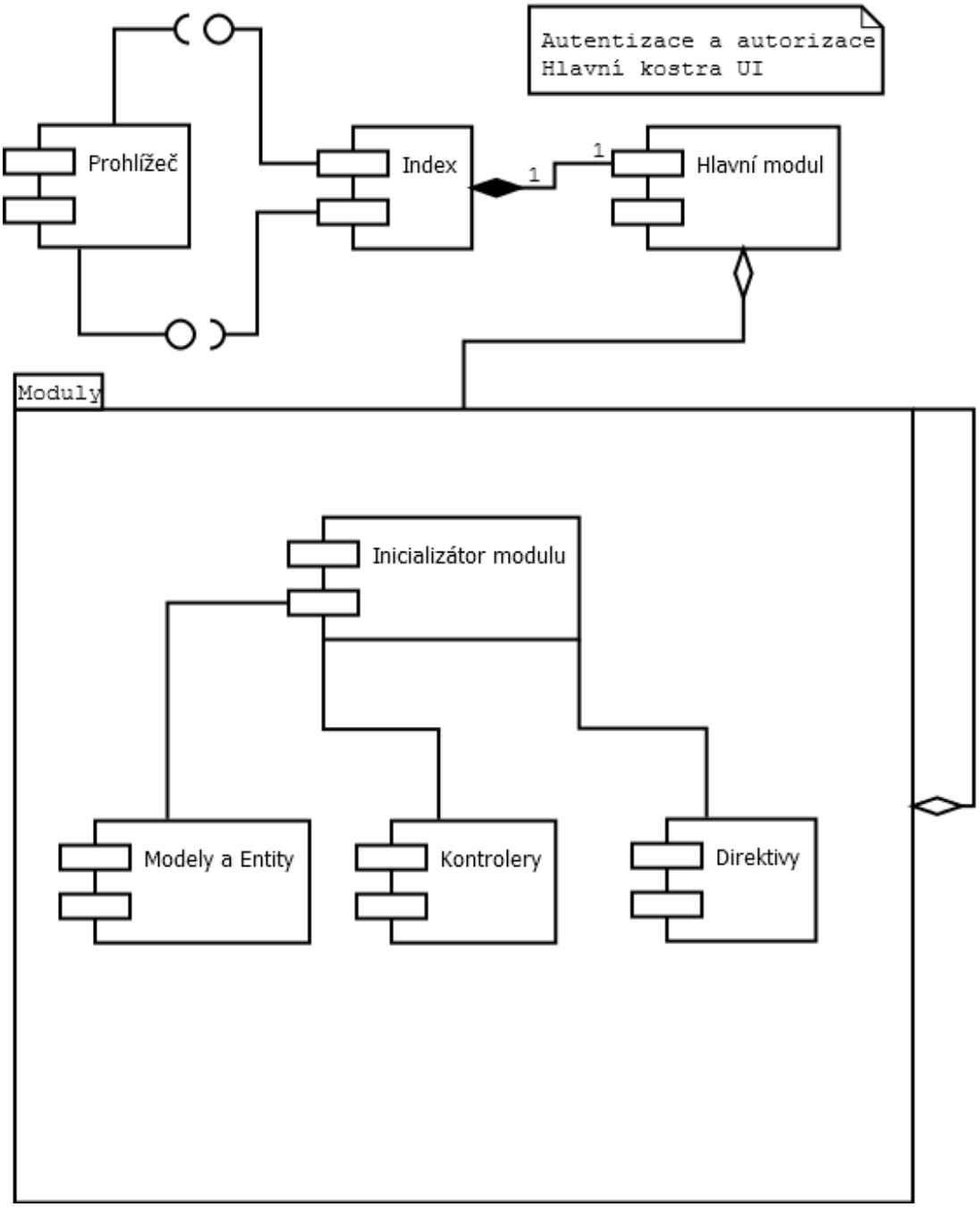
1. HTTP sloveso GET pro získávaní dat
2. HTTP sloveso POST pro vytváření dat
3. HTTP sloveso PUT pro aktualizaci dat
4. HTTP sloveso DELETE pro odstranění dat

Z hlediska zabezpečení bude použito ACL a pro identifikaci uživatele implementace protokolu OAuth nebo OAuth2. Dále je aplikaci nastaveno omezení pomocí HTTP hlavičky Access-Control-Allow-Origin.

4.2.2 Klient

4.2.2.1 Struktura Aplikace

Klientská část je napsaná v JavaScriptovém frameworku Angular.js s využitím TypeScriptu. Aplikace je členěna do hierarchické struktury modulárních a vzájemně nezávislých komponent.



Obrázek 3: Modularita Klienta skladového hospodářství

Uživatelské rozhraní je vytvořeno pomocí komponent angular material. Při vytváření UI jsem se zaměřil na logickou podobnost s procesem správy skladu a využití běžně používaných prvků, které usnadňuje a urychluje učení nových uživatelů systému.

4.2.2.2 Sestavovací nástroje

Použití TypeScriptu přidalo nutnost kompilace z TypeScriptu do JavaScriptu a následně při nasazení provést spojení a minimalizaci CSS a JS souborů. Tento problém jsem vyřešil zavedením sestavovacího nástroje webpack. Toto umožnilo další členění JS souborů a jejich následné skládání. Další výhodu, kterou použití webpacku přineslo, je integrovaný vývojářský server. Tento vývojářský server spouští aplikaci v iframu a pomocí websocketů a sledování zdrojových souborů provádí automatické znovunačtení aplikace. Webpack dále rozlišuje vývojářský a produkční režim a při vývojářském režimu přidává mapu z minifikováného javascriptu do zdrojových TypeScript souborů a tím značně usnadňuje ladění.

4.2.3 Další vývoj

Aplikace je ve stavu kostry s vývojovým prostředím. Dalším krokem vývoje bude dokončení základní funkcionality aplikace a následně implementace integrace s E-shopem. Dalším krokem je implementace komunikace pomoci websocketů, která umožní synchronizaci operací mezi více uživateli. Další vylepšení vhodné k implementaci je podpora více skladových strategií a implementace rozhraní pro integraci s dalšímy systémy klienta.

5 Závěr

Během trvání praxe jsem pomohl rozšířit funkcionalitu E-shopu, pomohl s návrhem systému pro automatizovanou instalaci a aktualizaci E-shopu a začal jsem implementaci řešení pro skladové hospodářství.

Vzhledem k prioritizaci úkolů směrem k rozšiřování funkcionalit E-shopu byly jen vytvořeny kostra skladového hospodářství a návrh pro implementaci automatizace instalace.

Z hlediska budoucího vývoje dojde k dokončení skladového hospodářství a následně k integraci E-shopu se skladovým hospodářstvím. Automatizace instalace bude nadále mít nízkou prioritu a práce na tomto úkolu bude využívat jen volný vývojový čas pokud nedojde k významnému zvýšení frekvence vydávání verzí.

Pro vylepšení vývojového cyklu a zjednodušení schvalovacího procesu po implementaci automatizované instalace by bylo vhodné rozšířit vývojový proces o tvorbu jednotkových (unit) a E2E testů, které sníží množství ručního testování provázející téměř každou změnu

Během praxe jsem zdokonalil svou znalost frameworku Nette a získal zkušenost s práci v malém týmu. Dále jsem byl seznámen s provozem E-shopu a strategiemi provozu skladového hospodářství. Dále jsem získal zkušenosti tvorbou UI, která je uživatelsky přívětivá a zároveň odolné proti uživatelským chybám.

Z hlediska technických schopností jsem s úkoly, zadanými v průběhu praxe, neměl žádný zásadní problém ale z hlediska znalostí pracovního postupu E-shopu a hlavně vedení skladového hospodářství jsem si musel doplnit znalosti.

Literatura

[1] Eparts Group s.r.o. Web společnosti Eparts Group s.r.o. [online]. Česká republika: Eparts Group, ©2012-2016 [cit. 2016-03-14]. Dostupné z: http://www.epartsgroup.cz/

[2] Mysql documentation: what is mysql. MySQL 5.7 Reference Manual [online]. Oracle Corporation and/or its affiliates, 2016 [cit. 2016-03-25]. Dostupné z: http://dev.mysql.com/doc/refman/5.7/en/what-is-mysql.html

[3] PHP: Hypertext Preprocessor. PHP: Hypertext Preprocessor [online]. The PHP Group, ©2001-2016 [cit. 2016-03-25]. Dostupné z: http://php.net/

[4] Nette Framework: zvyšte svoji produktivitu. Zdroják.cz [online]. Zdrojak.cz, 2009 [cit. 2016-03-25]. Dostupné z: https://www.zdrojak.cz/clanky/nette-framework-zvyste-svoji-produktivitu/

[5] Dibi: Quick Start. Dibiphp.cz [online]. Nette Foundation, 2016 [cit. 2016-03-25]. Dostupné z: https://dibiphp.com/cs/quick-start

[6] JQuery. Jquery.com [online]. The jQuery Foundation, 2016 [cit. 2016-03-25]. Dostupné z: https://jquery.com/

[7] Bootstrap: README. Github.com/twbs/bootstrap: Git repository of twbs/bootstrap [online]. 2015 [cit. 2016-03-25]. Dostupné z: https://github.com/twbs/bootstrap/blob/v3.3.6/README.md

[8] AngularJS: Developer Guide: Introduction. AngularJS [online]. © 2010-2016 [cit. 2016-03-27]. Dostupné z: https://docs.angularjs.org/guide/introduction

[9] Node.js. Node.js [online]. Node.js Foundation, 2016 [cit. 2016-03-27]. Dostupné z: https://nodejs.org/en/

[10] Restify: API Guide. Restify [online]. 2015 [cit. 2016-03-27]. Dostupné z: http://restify.com/

[11] Angular Material: Introduction. Angular Material [online]. Google, 2016 [cit. 2016-03-27]. Dostupné z: https://material.angularjs.org/1.0.6/

[12] EDEN, A a J NICHOLSON. Codecharts: roadmaps and blueprints for object-oriented programs. Hoboken, NJ: Wiley, 2011. ISBN 978-0-470-62694-8. Dostupné také z: http://www.eden-study.org/articles/2011/Codecharts.pdf

[13] TypeSript: JavaScript that scales [online]. Redmond: Microsoft, ©2012-2016 [cit. 2016-04-17]. Dostupné z: http://www.typescriptlang.org/